Multiway Trees

Two trees are said to be *isomorphic* if they "have the same shape"; that is, if there is a one-to-one correspondence between the nodes of the two trees that preserves the parent-child relationships of all of the nodes.

In this assignment, you will read in data to be stored in two multiway trees. You will then print out each tree in *preorder*, and then determine if the trees have the *same shape.*

Input:

The input for your program will be the file MultiwayTreeInput.txt. This file contains an even number of lines, each containing a Python list representation of a multiway tree.

Details about your program:

Your program should:

- Read in the first line and print it out.
- Convert the Python list representation into a node-and-pointer representation of the tree.
- Print out the value of the nodes in the node-and-pointer representation of the tree using *preorder.*
- Read in the second line and print it out.
- Convert the Python list representation into a node-and-pointer representation of the tree.
- Print out the value of the nodes in the node-and-pointer representation of the tree using *preorder.*
- Call the method "isIsomorphicTo()" to determine whether or not the two trees are isomorphic to each other, and print an appropriate message.
- Repeat this process for each pair of trees in the file.

Your program must look like the following:

```
     class MultiwayTree:

        def __init__(self,pyTree):  given "pyTree", a Python representation
of a tree, create a
             node-and-pointer representation of that tree.

        def preOrder(self):  print out the node-and-pointer representation of
a tree using preorder.

        def isIsomorphicTo(self,other):  return True if the tree "self" has
the same structure as the
             tree "other", "False" otherwise.
```

```
    def main():

    main()
```

The class MultiwayTree can also contain any additional access methods that you need, such setRootVal(), getRootVal(), etc.

Output:

Your program should produce output similar to the following:

```
   Tree 1:  [1, [[2,[]], [3, [ [5,[]], [6, [ [10,[]] ] ] ]], [4, [ [7,[]],
[8,[]], [9,[]] ]] ]]
   Tree 1 preorder:   1 2 3 5 6 10 4 7 8 9

   Tree 2:  ["A", [["B",[]], ["C", [ ["E",[]], ["F", [ ["J",[]] ] ] ]], ["D",
[ ["G",[]], ["H",[]], ["I",[]] ]] ]]
   Tree 2 preorder:   A B C E F J D G H I

   Tree 1 is isomorphic to Tree 2


   Tree 3:  ["Z", [], []]
   Tree 3 preorder:   Z

   Tree 4:  [26, [], []]
   Tree 4 preorder:   26

   Tree 3 is isomorphic to Tree 4


   Tree 5:  [1, [[2, [[5,[]]]], [3,[[6,[]],[7,[]]]], [4,[]]]]
   Tree 5 preorder:   1 2 5 3 6 7 4

   Tree 6:  ["A", [["B",[["E",[]]]], ["C",[["F",[]], ["G",[]]]],["D",[]]]]
   Tree 6 preorder:   A B E C F G D

   Tree 5 is isomorphic to Tree 6


   Tree 7:  [1, [ [2,[]], [3,[]] ]]
   Tree 7 preorder:   1 2 3

   Tree 8:  ["A", [ ["B", [ ["C",[]] ] ]]]
   Tree 8 preorder:   A B C

   Tree 7 is not isomorphic to Tree 8
```

```
   Tree 9:
[1,[[2,[[3,[[5,[]],[6,[[10,[]]]],[7,[]]]],[4,[[8,[]],[9,[[11,[]]]]]]]]]
   Tree 9 preorder:   1 2 3 5 6 10 7 4 8 9 11

   Tree 10:
["A",[["B",[["C",[["E",[]],["F",[["J",[]]]],["G",[["K",[]]]]]],["D",[["H",[]]
,["I",[]]]]]]]
   Tree 10 preorder:   A B C E F J G K D H I

   Tree 9 is not isomorphic to Tree 10


etc.
```